Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)

This listing of claims will replace all prior versions, and listings, of claims in the application:

What is claimed is:

1. (Currently amended) A system for translation of data types between a first application in a first language and a second application in a second language, the system comprising:

a computer having at least one processor and a memory;

a formal mapping between data types of the first language and data types of the second language;

translators for translating data types between the first language and the second language based on the formal mapping;

a translation mapping to the translators based on actual data types of the first application and formal data types of the second application; and

a module for automatically selecting an appropriate translator for translating between a particular data type in the first language and a data type in the second language based on the translation mapping in response to invocation of a method of the first application with the particular data type.

2. (Currently amended) The system of claim 1, wherein the first language comprises C# and the second language comprises ~~Java~~ a language other than C#.

3. (Original) The system of claim 1, wherein the formal mapping comprises a mapping between formal types of the first language and formal types of the second language.

4. (Original) The system of claim 3, wherein the formal types comprise static types.

5. (Original) The system of claim 1, wherein the formal mapping comprises a

many-to-one mapping.

6. (Original) The system of claim 1, wherein the translators marshal translated data into a wire format for transfer from the first application to the second application across a network.

7. (Original) The system of claim 1, wherein the translators read data of a first type and write data of a second type.

8. (Original) The system of claim 1, wherein the translators include a mechanism for determining the actual type in the first language that a particular translator supports.

9. (Original) The system of claim 1, wherein the translators include a mechanism for determining the formal type in the second language that a particular translator supports.

10. (Original) The system of claim 1, wherein the translators provide information needed for creating the translation mapping.

11. (Original) The system of claim 1, wherein the translators translate return values received from the second application into a format appropriate for the first application.

12. (Original) The system of claim 1, wherein the translation mapping provides for navigation from an object of the first application to a formal type of the second application's environment.

13. (Original) The system of claim 1, wherein the translation mapping comprises a mapping from actual type of the first application and formal type of the second application to a particular translator.

14. (Original) The system of claim 1, wherein the module for selecting an appropriate translator performs a two level lookup in the translation mapping.

15. (Original) The system of claim 14, wherein the two level lookup includes a first level lookup based on actual data type of the first application.

16. (Original) The system of claim 15, wherein the first level lookup considers inheritance hierarchy of the actual type.

17. (Original) The system of claim 14, wherein the two level lookup includes a second level lookup based on formal data type of the second application.

18. (Original) The system of claim 17, wherein the second level lookup selects the appropriate translator from a set of translators determined by the first level lookup.

19. (Original) The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for the particular data type.

20. (Original) The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for interfaces of the particular data type.

21. (Original) The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for base types of the particular data type.

22. (Currently amended) A method for translation of data types between a first component in a first language and a second component in a second language, the method comprising:

defining a formal mapping between data types of the first language and data types

of the second language;

implementing translators based on the formal mapping for translating data types between the first language and the second language;

producing a programming interface for the first component based upon the formal mapping and the second component's programming interface;

generating a translation mapping to the translators based on actual data types of the first component and formal data types of the second component as defined in the first component's programming interface;

in response to invocation of a method defined in the first component's programming interface with a particular data type, automatically selecting a translator based on the translation mapping and the particular data type; and

translating the particular data type to a data type of the second language using the selected translator.

23. (Original) The method of claim 22, wherein the first component comprises an application on a first machine and the second component comprises an application on a second machine.

24. (Original) The method of claim 22, wherein the first component comprises a first component of an application and the second component comprises a second component of the application.

25. (Original) The method of claim 22, wherein the first component and the second component operate within a single process.

26. (Original) The method of claim 22, wherein the defining step includes defining a mapping between formal types of the first language and formal types of the second language.

27. (Original) The method of claim 22, wherein the defining step includes defining a many-to-one mapping.

28. (Original) The method of claim 22, wherein the implementing step includes implementing a translator for marshaling translated data into a wire format for transfer from the first component to the second component across a network.

29. (Original) The method of claim 22, wherein the implementing step includes implementing a translator reading data of a first type and writing data of a second type.

30. (Original) The method of claim 22, wherein the implementing step includes indicating the actual type in the first language that a particular translator supports.

31. (Original) The method of claim 22, wherein the the implementing step includes indicating the formal type in the second language that a particular translator supports.

32. (Original) The method of claim 22, wherein the generating step includes generating the translation mapping based, at least in part, on information provided by the translators.

33. (Original) The method of claim 22, wherein the translation mapping provides for navigation from an object of the first component to the formal type of the second component's environment.

34. (Original) The method of claim 22, wherein the translation mapping comprises a mapping from actual type of the first component and formal type of the second component to a particular translator.

35. (Original) The method of claim 22, wherein the selecting step includes performing a two level lookup in the translation mapping.

36. (Original) The method of claim 35, wherein the two level lookup includes a

first level lookup based on actual data type of the first component.

37. (Original) The method of claim 36, wherein the first level lookup considers inheritance hierarchy of the actual type.

38. (Original) The method of claim 35, wherein the two level lookup includes a second level lookup based on formal data type of the second component.

39. (Original) The method of claim 38, wherein the second level lookup includes selecting a translator from a set of translators determined by the first level lookup based on formal data type.

40. (Original) The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for the particular data type.

41. (Original) The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for interfaces of the particular data type.

42. (Original) The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for base types of the particular data type.

43. (Original) The method of claim 22, further comprising:

translating return values received from the second component into a data type of the first component's environment using the selected translator.

44. (Currently amended) The method of claim 22, wherein the first language is C# and the second language is ~~Java~~ a language other than C#.

45. (Currently amended) The method of claim 22, wherein ~~the first language is Java and~~ the second language is C# and the first language is a language other than C#.

46. (Currently amended) The method of claim 22, further comprising:

copying to a ~~A~~ computer-readable medium ~~having~~ processor-executable instructions for performing the method of claim 22; and

executing said processor-executable instructions upon placement of the computer-readable medium in a computer.

47. (Currently amended) The method of claim 22, further comprising:

downloading a ~~A downloadable~~ set of processor-executable instructions for performing the method of claim 22; and

executing said processor-executable instructions in a computer upon completion of the downloading step.